An Approach for Detection and Family Classification of Malware Based on Behavioral Analysis

Steven Strandlund Hansen, Thor Mark Tampus Larsen, Matija Stevanovic and Jens Myrup Pedersen
Department of Electronic Systems
Aalborg University, Denmark
Email: {ssh, tmtla, mst, jens}@es.aau.dk

***Abstract*—Malware, i.e., malicious software, represents one of the main cyber security threats today. Over the last decade malware has been evolving in terms of the complexity of malicious software and the diversity of attack vectors. As a result modern malware is characterized by sophisticated obfuscation techniques, which hinder the classical static analysis approach. Furthermore, the increased amount of malware that emerges every day, renders a manual approach inefficient. This study tackles the problem of analyzing, detecting and classifying the vast amount of malware in a scalable, efficient and accurate manner. We propose a novel approach for detecting malware and classifying it to either known or novel, i.e., previously unseen malware family. The approach relies on Random Forests classifier for performing both malware detection and family classification. Furthermore, the proposed approach employs novel feature representations for malware classification, that significantly reduces the feature space, while achieving encouraging predictive performance. The approach was evaluated using behavioral traces of over 270,000 malware samples and 837 samples of benign software. The behavioral traces were obtained using a modified version of Cuckoo sandbox, that was able to harvest behavioral traces of the analyzed samples in a time-efficient manner. The proposed system achieves high malware detection rate and promising predictive performance in the family classification, opening the possibility of coping with the use of obfuscation and the growing number of malware.**

***Keywords*—Malware, Dynamic Analysis, Malware Detection, Family Classification, Feature Selection, Random Forests**

I. Introduction

The Internet usage has significantly increased over the last decade. Private users, companies and governments rely more and more on Internet-based services, which as a consequence make them increasingly vulnerable to cyber attacks or infections by malicious software a.k.a malware. In parallel with the increasing Internet usage, a lucrative black market has developed, where one can purchase malicious software and/or other malicious services. In order to take the full advantage of the black market cyber criminals do their best to increase the complexity of their malicious code in order to hide and obfuscate it from known detection methods. Obfuscation can consist of different techniques such as: polymorphism, metamorphism and packing. This naturally leads to many different versions or new implementations of the functionalities found in each specific family of malware. Malware family represent a group of malicious programs that usually originates from the same source code or share a common behavior. In terms of malware infections, AV-test registers approximately 390,000 new malicious programs every day [1]. Furthermore, their statistics denote a three-fold increase in the amount of new malware from 2012 to 2014 [1]. As a result, it has become problematic to manually process data obtained by analyzing such a vast amount of malware samples, which can make it challenging for anti-virus companies to detect and classify the malware. This means that it is increasingly difficult to prevent infections from novel malware, and release timely updates.

To circumvent the problem of handling malware obfuscation and the growing number of malicious software, recent research aims to find a more efficient method of malware detection and family classification, that can overcome the limitations of manual static analysis [2]–[6]. In this paper, we build on top of the concepts and findings presented in the previous work proposing a novel scalable dynamic analysis approach. The proposed approach has a goal of detecting malware and classifying it into appropriate malware families. In order to achieve so we perform dynamic analysis of malware by injecting captured malware samples into parallel virtual environments and recording their actions performed on the operating system. Furthermore, we define binary classifier consisting of "malware" and "cleanware" (benign software), assuming that it is possible to detect malicious programs based on the behavioral data retrieved from the dynamic analysis. Additionally, we developed a multiclass classifier using malware family-based classes, which can provide a more detailed picture of malware behavior. The novel approach has been evaluated using one the most extensive set of malware and clenware samples of over 270,000 and 837 samples, respectively. The proposed method has proven to be both efficient and scalable, compared to other research articles [2]–[6], being only limited by the provided hardware. Finally, it should be noted that work presented in this paper builds on top of our previous work [7] by utilizing both malware detection and family classification based on a novel feature representations.

The rest of the paper is organized as follows. Section II presents an overview of related work. Section III introduces a novel approach for malware detection and family classification. Section IV presents the results of performance evaluation. Section V discusses presented results and possibilities for future work. Finally, Section VI concludes the paper.

II. Related Work

In this section we present the state-of-the-art approaches that use dynamic analysis to perform either malware detection or family classification. We focus on methods that rely on supervised machine learning as the tool of classifying malware. We do so as many existing work rely on supervised machine

978-1-4673-8579-4/16/$31.00 ©2016 IEEE

learning for discriminating between malware families, based on the analysis of behavioral data.

Fukushima et al. [2] have proposed detection method with the goal to detect malware without any false positives, based on malware and cleanware behavior. The authors used 83 malware and 41 cleanware for evaluation, achieving 60% accuracy of malware detection with no false positives. Salehi et al. [3] and Uppal et al. [4] have proposed methods that focused on API calls applied as behavioral features in machine learning to detect malware. Uppal et al. [4] represented API calls according to the sequence in which they were called. Salehi et al. [3] used a combination of API calls and their input arguments. The authors also used feature selection techniques to lower the amount of features, making it possible to use around 2,000 features when building the classifier. Both groups of authors used around 1,000 malware and cleanware in total for testing, reporting an accuracy of around 98%.

Tian et al. [5] proposed a method for both malware detection and family classification, where API calls were used as features to describe the malware behavior. In this paper a frequency representation for API calls was used. A total of 1,368 malware and 456 cleanware were used for evaluating several machine learning algorithms. An accuracy of approximately 97% was achieved for both detection and classification utilizing the best performing algorithms. Rieck et al. [6] proposed framework, that can handle thousands of malware each day in an automated manner applying the behavior of malware in machine learning. Finally, in our previous work [7] we proposed a method for malware type classification that indicated promising possibilities of classifying malware based on malware behavioral analysis.

In this paper we follow the principles of malware detection and family classification used by the existing work. Our contribution lies in the choice of features and their representations, along with the reduction strategy performed on the feature space. Furthermore, we have evaluated the proposed approach using one of the most extensive data sets of malware and cleanware, in comparison to the aforementioned papers. The features used in this study are API calls along with specific input arguments, and behavior signatures retrieved from AV-vendors' malware encyclopedia. The representation of these features, include a combination of sequence, frequency and binary techniques. This representation technique together with the features and reduction technique, is meant to give a better predictive performance, compared to only using one technique with one kind of feature, which does not cover enough of the behavior performed by the malware, leaving less data for discrimination. Information Gain Ratio is used as a relevance measure to perform feature selection, and thereby remove potential redundant and irrelevant features. Our setup is automated and scalable, enabling a large amount of malware to be analyzed for detection and classification.

III. Detection & Classification Method

To solve the problems introduced in the aforementioned sections, we have developed a novel approach for malware detection and family classification. This section explains the utilized methodology covering three important areas depicted in Figure 1, namely: *Data generation*, *Data Extraction* and *Detection & Classification*.

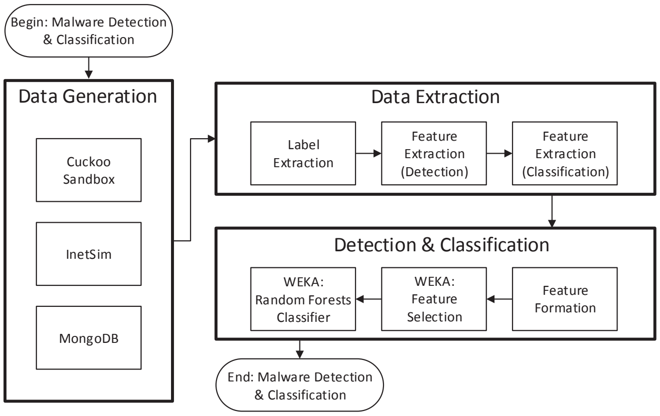

Begin: Malware Detection & Classification
Data Generation
Cuckoo Sandbox
InetSim
MongoDB
Data Extraction
Label Extraction
Feature Extraction (Detection)
Feature Extraction (Classification)
Detection & Classification
WEKA: Random Forests Classifier
WEKA: Feature Selection
Feature Formation
End: Malware Detection & Classification

Fig. 1. Overall system flow for both Malware Detection and Classification.

A. Data Generation

In order to obtain behavioral traces of malware and cleanware samples we have developed a scalable and distributed malware analysis setup based on Cuckoo Sandbox [8]. Cuckoo Sandbox represents a powerful malware analysis systems that is able to trace various client-level forensics by executing malware within Virtual Machines (VMs). In order to analyze a large amount of samples efficiently, we modified Cuckoo to work in a scalable and distributed manner. This was done such that commands to control the VMs, were send over a closed network using Secure Shell (SSH), making it possible to connect multiple machines as VM controllers and distribute the analysis. In our setup we used 30 VMs distributed over several physical machines. Furthermore, we deployed a server to collect all the analysis data from the VMs and store it in MongoDB, a NoSQL database. During analysis, INetSim [9] has been used to emulate Internet services, in order for the VMs to act as normal computers. On each VM, Windows 7 with no service pack was installed together with personalized and commonly used programs. Finally, the samples were analyzed for up to 200 seconds, which we believe is sufficient time to provide us with enough behavioral data while maintaining efficiency of the setup. Malware evasion techniques were not considered in this project.

B. Data Extraction

Data extraction is used to extract labels and features, for each of the malware and cleanware samples. Labels for malware families were extracted from VirusTotal [10] reports generated for each malware sample. It should be noted that we use labels according to Microsoft Security Essentials (MSE). The features are based on API calls and their input arguments. The input arguments from API calls, consist of strings, mutexes, DLLs, registry keys, etc. We have considered two different feature representations, as illustrated in Figure 2. The main difference between the two representations is the way the sequence is defined. There is also a slight difference between the representations used for the malware detection and the family classification. This is due to what we denote as *"signature-based"* features, which are only present for the families. The full feature sets have a specific length, but due to feature selection, only a subset of the feature representation are used, including the most discriminative features.

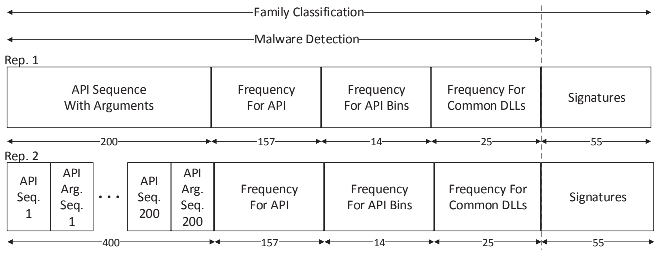


Fig. 2. Feature Representations.

1) Malware Detection: Malware detection represents binary classification problem with two classes, namely "malware" and "cleanware". The assumption is that a behavioral difference is present for the two classes, making it possible to discriminate between malicious and benign behavior. For each malware and cleanware sample a set of discriminative features is extracted. The feature set consists from Sequence and Frequency features implemented in two distinct feature representations i.e. Representation 1 and Representation 2.

Sequence: The sequence is the order in which API calls, including their arguments, are called on the VM. Two different representations have been made for sequence features. In Representation 1, the first 200 APIs are listed in the order in which they are called, where each API and its input argument are combined into one feature. In Representation 2, the API and its input argument are separated, such that 200 extra features are introduced. The input arguments used in this representation are only the ones that are connected with calls to new functions or APIs, else a "0" will be introduced.

Frequency: The frequency is the number of occurrences of API calls during the analysis on the VM. For both feature representations, the same frequency techniques are used. For each API call a frequency is assigned in the representation. Furthermore, feature construction technique using the addition operator, produces 14 different behavioral API bins, where frequency for each API in the corresponding bin are summed. Finally, frequencies of the 25 most used DLL files are used.

2) Family Classification: The family classification represents multiclass classification problem where number of classes is equal to the number of families. The family classification uses the same representation techniques as for the malware detection, with the addition of 55 signature-based features. The idea is to use commonly known signatures extracted from AV-vendors' malware encyclopedia for each malware family [11]. It is assumed that in the best case scenario, each signature only belongs to one family, which provide us with very powerful discriminative features.

Binary: The signature-based features represent binary features indicating if the signature is present in the API arguments, or not. An example signature could be from the malware family called Hupigon, which is commonly known to use specific strings when performing operations on the compromised machine. Here, it has been confirmed that it uses strings like: `hacker.com.cn.exe`.

C. Detection & Classification

This section presents the applied Feature Selection (FS) algorithm and the used classification algorithm.

1) Feature Selection: Data based on dynamic analysis might provide potentially irrelevant or redundant features, thus the use of FS algorithms can be very beneficial. This study uses the concept of Information Entropy (IE), where Information Gain Ratio (IGR) is used as the relevance measure. This measure is defined by the IE, in following:

$$IGR(C,F) = \frac{H(C) - H(C|F)}{H(F)} \tag{1}$$

where $H(C)$ and $H(F)$ denote the class and feature distribution, respectively while $H(C|F)$ denotes the class distribution, given the feature distribution. Using this relevance measure, it is believed that irrelevant and redundant features can be filtered [12]. This is important, since these degrade the predictive performance of the used classifier.

2) Random Forests: Random Forests (RF) classifier [13] is used for both malware detection and family classification. RF is an ensemble classifier that relies on a multitude of trees to reduce the variance in the classification and thereby improve the predictive performance. Using the majority vote of the predictions from multiple different trees generated by Bootstrapping, RF obtains a prediction model, that has reduced risk of over-fitting. In this study, each sample has been injected into a number of trees, whereas a decision has been made at each node, based on the Information Gain of the best node-split among a random subset of features, contained in the feature space. For the implementation of the classifiers for both malware detection and family classification we use RF classifier with 50 trees.

IV. Evaluation

Data Extraction and Detection & Classification are implemented in Python and work as serialized modules, while we rely on pre-implemented algorithms from the Machine Learning toolbox WEKA [14]. For the evaluation we use behavioral traces from approximately 270,000 malware and 837 cleanware samples [15]–[18].

A. Results

The performance was evaluated using using typical supervised machine learning scheme where 67% of the samples was used for training, whereas testing was realized using the remaining 33%. Furthermore, we evaluated the performance in regards to the used feature representation. For malware detection and family classification, Representation 2 and 1 performed best, respectively. Malware classification was evaluated using total sample set containing 5,000 malware and 837 cleanware samples. It should be noted that each malware samples belonged to distinct malware families. Family classification was implemented for five classes (families), namely Small, OnlineGames, Hupigon, Frethog and Zlob. The five classes correspond to a total of 31,295 malware samples. The remaining families were not used due to insufficient amount of samples for each particular class in the available data set.

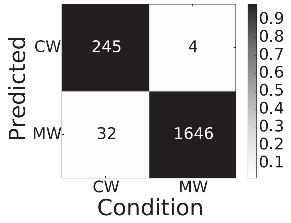
The results are described using the following performance metric: True Positive Rate (TPR), False Positive Rate (FPR), Positive Predictive Value (PPV), F-measure (F-M) and Area Under the Curve (AUC) of the Receiver Operating Curve.

1) Malware Detection: Malware detection performed the best for Representation 2. Here, the use of FS reduced the amount of features to only 50, which is a reduction of 88%. The performance of malware detection is presented in Table I. The performance of malware classification is characterized with a high F-M, due to high TPR and PPV. Furthermore, malware classification is characterized by high FPR due to the high number of False Positives (FPs) and low number of cleanware samples. Cleanware classification shows slightly lower F-M than in the case of malware classification. Overall, from the weighted average, the performance of the malware detection is satisfactory, because of the high TPR and PPV.

TABLE I. PERFORMANCES OF MALWARE DETECTION.

Class	TPR	FPR	PPV	F-M	AUC
Cleanware	0.884	0.002	0.984	0.932	0.996
Malware	0.998	0.116	0.981	0.989	0.996
Weighted Avg.	0.981	0.099	0.981	0.981	0.996

Digging deeper into the individual predictions, a confusion matrix is depicted in Figure 3. Here, it is noticed that cleanware had only 4 FPs out of the 1,650 tested malware samples. This is also the reason for the high PPV for cleanware, which conclusively shows that the detection of malware works. The number of FNs for cleanware, show that 32 cleanware were predicted as malware. It should be noted that when Representation 1 was used to build the classifier, one more FP and 22 more FNs, were seen for cleanware. Therefore, a significant improvement was seen when API calls and input arguments were separated.



	CW	MW
CW	245	4
MW	32	1646

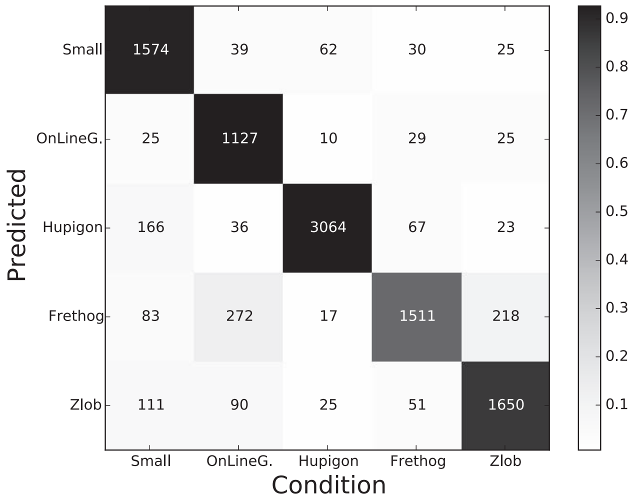
Fig. 3. Confusion matrix for malware (MW) / cleanware (CW) detection with Representation 2.

2) Family Classification: Family classification shows the best predictive performances for Representation 1 with 400 features after FS. Studying the results in Table II for the five classes, the best performance is obtained for Hupigon. This might be due to the largest amount of samples, and many signature-based features. Frethog had a lower performance for FPR and PPV, which might be due to fewer discriminative features, leaving it less defined. Overall, from the weighted average, the performance of the family classification is promising, because of the relatively high TPR and PPV.

TABLE II. PERFORMANCES OF FAMILY CLASSIFICATION.

Class	TPR	FPR	PPV	F-M	AUC
Small	0.803	0.019	0.910	0.853	0.972
OnlineGames	0.721	0.010	0.927	0.811	0.968
Hupigon	0.964	0.041	0.913	0.938	0.990
Frethog	0.895	0.068	0.719	0.798	0.972
Zlob	0.850	0.033	0.856	0.853	0.979
Weighted Avg.	0.864	0.035	0.872	0.864	0.978

Analyzing individual predictions in more detail, a confusion matrix is depicted in Figure 4. Studying the FNs for Small, it is seen that many of these are predicted as Hupigon and Zlob, which also explains its lower TPR. Looking at the performance for OnlineGames, few FPs are observed, but a high amount of FNs are noticed as 272 samples of OnlineGames are classified as Frethog. Hupigon has the highest performance, without any significant FNs or FPs, with the exception for the FPs for Small. Frethog has a low amount of FNs, but many FPs where 272 samples of OnlineGames and 218 samples of Zlobs are wrongly classified as Frethog, leaving it with the lowest PPV. At last, Zlob has a low amount of FPs and FNs for most of the classes, except for its FNs for Frethog and FPs for Small.



	Small	OnLineG.	Hupigon	Frethog	Zlob
Small	1574	39	62	30	25
OnLineG.	25	1127	10	29	25
Hupigon	166	36	3064	67	23
Frethog	83	272	17	1511	218
Zlob	111	90	25	51	1650

Fig. 4. Confusion matrix for family classification with Representation 1.

V. DISCUSSION

This section elaborates on the performance of the proposed approach for malware detection and family classification and opportunities for future work.

A. Malware detection

As all 5000 malware samples used for the evaluation of malware detection originate from different families, the results indicate that the proposed approach could perform well even on malware samples from unknown malware families. This indicates a generalized classifier that should work with other variants of malware as well. The detection approach performed worse on cleanware samples and we believe that this can be attributed to the number of available cleanware samples used for training of the classifier. However, the evaluation gave fewer FNs and FPs than expected, which indicates that the features chosen in FS, are overall good. Judging by the evaluation results, only 4 out of 1,650 malware samples were not detected, which conclusively illustrates the capabilities of the proposed detection approach. Improvements can be done in terms of achieving fewer FNs for cleanware, thus improving TPR. Overall, the performance is promising and using a more uniform sample set with higher number of cleanware samples, can improve the results.

B. Family classification

Studying the FNs, it is seen that many Smalls are predicted as Hupigon and Zlob. Small generally consists of Trojans, that connect to remote servers, whereas Hupigon is known as a Remote Access Tool. Zlob consists of multiple components, but are mostly known for hijacking the users browser, but has also been observed to act as a Trojan downloader, which means it has to connect to a remote server as well [11]. This concludes, that Small has functional similarities with Hupigon and Zlob. Frethog has many FPs, mainly to OnlineGames, where both target theft of user account credentials for online games, which might explain the miss-predictions. Many FPs for Zlob are also observed, since it is a very generic family and potentially consists of a wide variety of functionalities [11]. Generally, the results for the family classification are promising, where only few miss-predictions are present. A large sample set of malware was collected during this study, but more samples are needed to perform a large multiclass classification, where more significant number of families would be included. As the system is scalable and only limited by the provided hardware, it is possible to increase the number of classes, while maintaining a reasonable prediction performance. It was also noticed that Frethog had a tendency of having many FPs, which indicates, that more discriminative features are needed. Investing more time in finding and extracting better signature features, can increase the predictive performance while also coping with a larger class set.

C. Future Research

For malware detection, a more uniform sample set should be used by injecting more cleanware to the system. Also, more malware can be used to build the classifier, making it more robust. For family classification, better discrimination between some of the classes should be done, by utilizing more discriminative features. If all the different families are learned, it should be possible to use this setup to detect novel malware, that does not belong to any of the known families. Finally, the influence of malware analysis time to the observed malware behavior should be analyzed in order to avoid malware evasion and optimize the performance of the large-scale analysis.

VI. Conclusion

This study proposes a novel malware detection and family classification approach based on dynamic analysis that can detect malware, and classify them into appropriate families. The proposed approach relies on Random Forests classifier and novel feature representations for malware detection and family classification that significantly reduces the feature space, while achieving promising predictive performance. The approach has been evaluated using behavioral traces obtained by running over 270,000 malware and 837 cleanware samples within a malware testing environment based on Cuckoo sandbox. Using the proposed setup and available data sets, the proposed approach was able to achieve malware detection with weighted average TPR and PPV of 0.981 and 0.981, respectively. The results show potential for further development and real-time detection on client computers. Furthermore, the family classification achieved a weighted average TPR, PPV and AUC of 0.860, 0.867 and 0.977, respectively. It was noticed, that the family classification had minor trouble discriminating some of the families, indicating the need for more discriminative features in terms of e.g. signatures, to resolve the miss-predictions. Additionally, it is believed that the presented approach can be expanded to include a larger number of malware families, by providing a substantial number of malware samples. This would also make it possible to identify novel families in future work.

Acknowledgments

We would like to thank VirusTotal for providing us with the access to their malware analysis setup in order to obtain malware family labels from 56 anti-virus programs. Furthermore, we would like to thank Virus Share and VX Heavens, for providing us with the access to their extensive library of malicious software.

References

[1] AV-test, "New malware," http://www.av-test.org/en/statistics/malware/, July 2015.

[2] Y. Fukushima, A. Sakai, Y. Hori, and K. Sakurai, "A behavior based malware detection scheme for avoiding false positive," *Secure Network Protocols (NPSec), 2010 6th IEEE Workshop on*, pp. 79–84, October 2010.

[3] Z. Salehi, M. Ghiasi, and A. Sami, "A miner for malware detection based on api function calls and their arguments," *Artificial Intelligence and Signal Processing (AISP), 2012 16th CSI International Symposium on*, pp. 563–568, May 2012.

[4] D. Uppal, R. Sinha, V. Mehra, and V. Jain, "Malware detection and classification based on extraction of api sequences," *Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on*, pp. 2337–2342, September 2014.

[5] R. Tian, R. Islam, L. Batten, and S. Versteeg, "Differentiating malware from cleanware using behavioural analysis," *Malicious and Unwanted Software (MALWARE), 2010 5th International Conference on*, vol. 5, no. 5, pp. 23–30, 2010.

[6] C. W. K. Rieck, P. Trinius and T. Holz, "Automatic of analysis of malware behavior using machine learning," *Journal of Computer Security*, vol. 19(4), pp. 639–668, 2011.

[7] R. Pirscoveanu, S. Hansen, T. Larsen, M. Stevanovic, J. Pedersen, and A. Czech, "Analysis of malware behavior: Type classification using machine learning," in *Cyber Situational Awareness, Data Analytics and Assessment (CyberSA), 2015 International Conference on*, June 2015.

[8] Cuckoo, "Cuckoo," http://www.cuckoosandbox.org/, October 2014.

[9] T. Hungenberg and M. Eckert, "Inetsim: Internet services simulation suite - documentation," http://www.inetsim.org/documentation.html, November 2014.

[10] VirusTotal, "Virustotal : Free online virus, malware and url scanner," https://www.virustotal.com/en/documentation/, January 2014.

[11] Microsoft, "Microsoft threat encyclopedia," http://www.microsoft.com/security/pc-security/malware-families.aspx, July 2015.

[12] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, *Feature Extraction: Foundations and Applications*. Springer, 2006.

[13] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[14] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009.

[15] VX Heavens, "Vx heavens snapshot (2010-05-18)," https://archive.org/details/vxheavens-2010-05-18, 2010.

[16] VirusShare, "Virusshare 80k samples," http://tracker.virusshare.com:6969/torrents/VirusShare_00139.zip.torrent?CF174A50911F0AD662849A121A1D0D7C18140C8D, 2014.

[17] Ninite, "Ninite," https://ninite.com/, 2015.

[18] Lupo PenSuite Collections, "Lupo pensuite collections," http://www.lupopensuite.com/collection.htm, 2015.